RECEIVED
CENTRAL FAX CENTER
DEC 1 5 2006

REMARKS

Claims 1-30 are pending. The Examiner has maintained the 35 U.S.C. 112 rejection of the phrase "substantially all multi-byte aligned branch instructions." The Examiner has rejected claims 1-11, and 13-29 under 35 U.S.C. 102(b) as being anticipated by Intel (IA-32® Architecture Software Developer's Manual, Volumes 1-2, 2002) and has also cited the Intel article "Data Alignment when Migrating to 64-Bit Intel Architecture."

The Examiner objected to claims 1-13 under 35 U.S.C. 112(2) as being indefinite because of the phrase in independent claim 1 "substantially all multi-byte aligned branch instructions." The language in independent claim 1 and the use of the term "substantially" is believed proper. One of skill in the art would recognize what branch instructions are in light of the specification. For example, branch instructions include "branch on equal or branch on not equal, and jump instructions." (Page 6, lines 7-8) "The branch instructions such as branch on equal or branch on not equal instructions allow a change in the flow of execution of a program by jumping to a nonsequential instruction at a branch target address." (page 6, lines 27-29) One of skill in the art would recognize that an instruction set typically includes a "group of multi-byte aligned branch instructions." The group member can vary depending on the particular architecture. For example, branch instructions may include multi-byte aligned branch on equal, branch on not equal, branch on greater than, branch on less than, branch if equal to 0, branch based on contents of a register, unconditional jump, jump to a register address, etc.

Claim 1 includes the recitation "substantially all multi-byte aligned branch instructions." Requiring all multi-byte aligned branch instructions is believed to be inappropriate, as a potential infringer could easily design around by adding an extraneous multi-byte aligned instruction or a multi-byte aligned test instruction that does not perform claim recitations. The extraneous instruction may be completely frivolous, but would prevent infringement of a claim requiring something of "all multi-byte aligned branch instructions." One of skill in the art would understand the phrase "substantially all multi-byte aligned branch instructions." It is not necessary to define a particular threshold (e.g. 23 out of 25 multi-byte aligned instructions or 92%) as meeting the criteria for substantially. See *Andrew Corp. v. Gabriel Electronics*, 847 F.2d 819, 6 USPQ2d 2010 (Fed. Cir. 1988).

The Examiner has rejected claims 1-11, and 13-29 under 35 U.S.C. 102(b) as being anticipated by Intel (IA-32® Architecture Software Developer's Manual, Volumes 1-2, 2002) and has also cited the Intel article "Data Alignment when Migrating to 64-Bit Intel Architecture."

The Examiner argues that the Intel article "Data Alignment when Migrating to 64-Bit Intel Architecture" states that aligning data correctly can be an important optimization, although its use is still optional. More specifically, "One of these is the alignment of data items – their location in memory in relation to addresses that are multiples of four, eight or 16 bytes. Under the 16-bit Intel architecture, data alignment had little effect on performance, and its use was entirely optional. Under IA-32, aligning data correctly can be an important optimization, although its use is still optional with a very few exceptions, where correct alignment is mandatory." The Applicants respectfully disagree that this teaches or suggests the recitation "wherein substantially all multi-byte aligned branch instructions are operable to access the instructions at byte aligned addresses."

It is acknowledged in the specification that many architectures do not require particular alignment. The article merely points out that data alignment was optional. However, the article does not teach or suggest that "substantially all" multi-byte aligned branch instructions (or even one multi-byte aligned instruction for that matter) are operable to access the instructions at byte aligned addresses. As noted in the specification, conventional devices provided different sets of branch instructions for different alignments. For example, one set of branch instructions would be operable to access instructions at word aligned addresses, e.g. jump32. Another set of branch instructions would be operable to access instructions at half-word aligned addresses, e.g. jump16.

One rationale for even using a jump32 instruction when a jump16 instruction is available is that a jump32 instruction allows access to a greater range of memory addresses or a larger memory space. "In general, it will be appreciated that restricting alignment to multi-byte or word aligned memory locations allows access to a greater range of memory addresses. More specifically, a limited number of bits are available to store the offset associated with a branch. In one simplified example, three bits or eight different values of offset are provided. If no multi-byte alignment is used, the branch target address is restricted to a three bit range of eight bytes in the memory space. However, if word alignment is used, the offset is multiplied by 4. The branch target address range is now expanded to a five bit range or thirty-two bytes in the memory space. In this example, the target address still can only be in 18 memory locations, however the eight specific locations are spread over a range of thirty-two bytes instead of being limited to a range of eight bytes." (page 9, lines 12-23)

In devices such as ASICs or Intel processors, it is less expensive to create a whole new set of branch instructions than it is to modify multi-byte branch instructions to access instructions at byte aligned addresses. "By adding a whole set of new branch instructions, additional subcircuitry will have to be introduced, potentially increasing hardware costs." (page 8, lines 7-9) "In many processors and ASICs, adding logic and subcircuitry to handle unique instructions is less-expensive than adding logic in a programmable chip to handle unique instructions." (page 9, lines 28-30)

Independent claim 14 recites "common subcircuitry is used to process the immediate field associated with one or more branch instructions and one or more non-branch instructions" and independent claims 20 and 27 recite "common subcircuitry operable to calculate a byte-aligned address, wherein the common subcircuitry is also configured to perform nonbranch operations." Intel does not describe common subcircuitry recited in the independent claims 14, 20, and 27. The Examiner argues that the common subcircuitry is an ALU, which is inherent. The Applicants respectfully disagree. An ALU is not used to perform branch or nonbranch operations. Intel does not teach or suggest or otherwise describe any common subcircuitry because Intel only describes a conventional IA-32 processor, which uses different groups of branch instructions and different circuitry to handle 8-bit versus 16-bit versus 32-bit instructions. As noted in the present specification, "one group of branch instructions handles 16-bit aligned instructions. Another group of similar branch instructions handles 8-bit aligned instructions. These redundant groups of branch instructions in conventional processors are highly inefficient." (page 8, lines 15-17)

In light of the above remarks, the rejections to the independent claims are believed overcome for at least the reasons noted above. Applicants believe that all pending claims are allowable in their present form. Please feel free to contact the undersigned at the number provided below if there are any questions, concerns, or remaining issues.

Respectfully submitted,
BEYER WEAVER & THOMAS, LLP

[illegible]

Godfrey K. Kwan
Reg. No. 46,850

P.O. Box 70250
Oakland, CA 94612-0250
(510) 663-1100